# Camera movement

**Keyboard handling**

Whenever one of the special keys (w,a,s,d, arrows, …) is pressed, the code of the key is saved in an object, so also multiple keys can be pressed at one time.

For back and forth (key W and S) the speed variable “backAndForth” is set to a positive or negative value. Each rendering cycle the elapsed time is multiplied with this value, which is used to calculate the new (x,y,z)-value with which the viewMatrix is translated with.

The same principle is used for left, right and up, down. A variable is set to a positive or negative value and each rendering cycle, a new value is calculated with which the viewMatrix is rotated.

**Mouse handling**

At mouse-button-down the position is saved. Then when moving the mouse, every difference (delta value) is calculated and is added to pitch and yaw (our variables for rotating the viewMatrix) until the mouse-button is released. (mouse-button-up event)  
  
**Animated camera flight**

For our animated camera flight we save the important data (duration, newPosX, newPosY, newPosZ newPitch, newYaw, stay) in a queue and whenever one queue element is executed, the next one is loaded and executed.

This algorithm is pretty simple. The first time from the animated camera flight, we need to load the first queue element and calculate the delta values of the new position (where the camera should move). These delta values are calculated by subtracting the current position from the new position and dividing it by the duration. When this is done, in every render cycle the delta is multiplied by the elapsed time to get the new interpolated position.

# Scene Graph Elements

**Manually Composed Model**

The manually composed models are the two snipers, the soldiers and the enemies in their camp. They simply consist of a set of cubes: two feet, two ‘under legs’, two ‘upper legs’, two under arms, two upper arms, a neck, a body and a height.

Each side of the cube consists of six vertices and each vertex additionally has a normal vector and a texture coordinate. The reason why we don’t used indices is because you need three normals and texture coordinates for each vertex.

**Animations of these Models**

There are two different animations. The enemies and the soldiers are able to walk. Therefor their legs and arms swing back and forth. The two snipers lie on the ground and crawl to their spot, where they have a free field of view.

To make the rotation of the body parts easier the rotation points of the cubes are on the top-right side. So there is no additional translation necessary.

# Materials

**At least two different Materials**

Actually we have three materials. The watch towers are brown, the tents white-grey and the bullets are gold. The implementation was pretty easy. We just had do use a MaterialSGNode.

# Textures

**Apply texture to self-made model**

We have applied a camouflage texture to our soldiers. Therefore we implemented an own SGNode that derives from AndvancedTextureSGNode. The problem of this node was that it set no uniform that you were able to distinguish if the current vertex belongs to a texture or not.

# Illumination

**Multiple Light Sources**

We actually have two light sources, one general one and one light source for the fire. The second light source therefor has the same color like the particles of the fire.

**Phong Shading**

We implemented it like we had done in the exercise lecture.